Development of the StratWar wargame software

Timothy J. Halloran[1]

StratWar, a computerized strategic nuclear wargame, developed for the Air Command and Staff College faced serious political and technical challenges during its development. It survived to become a success that was voted as the "Highlight of the Academic Year" by 500 Air Command and Staff College students and faculty who played the game in its debut at the Air Force Wargaming Center in spring of 1990. This is an account, including a narrative and a critical analysis, of the 18-month development of the wargame written by a member of its programming team.

The narrative presents the issues tackled by the development team throughout the project's lifetime. What were the up-front manpower, schedule, and technical constraints on the project? What problems, both technical and political, did the development team face? How did they solve them? How was a large and inadequately understood legacy simulation integrated into the wargame design? How was off-the shelf software selected and used? How did the team prepare for and execute the first operational use of the software? The narrative is followed by an analysis that presents ten lessons extracted from a critical examination of what went right and what went wrong during the project.

The Air Force Wargaming Center

The Air Force Wargaming Center (AFWC)[2] stages exercises where Air Force officers are educated through the use of simulation, known as wargames, to further their understanding of and experience in war-like situations [1]. Computer simulation is used as one tool to accomplish this objective. The AFWC is located on Maxwell Air Force Base in Montgomery, Alabama in the center of the four major schools hosting Air Force officer professional military education (PME). From 1988 to 1991, I worked as a software developer within this organization. At my arrival into the organization, the transition to a new computer-based wargaming system was encountering difficulties.

---

[1] The views expressed in this document are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

[2] The Air Force Wargaming Center is today known as the Air Force Wargaming Institute (AFWI).

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
25 FEB 2004	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Development of the Stratwar Wargame Software (A Practicum)	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
Timothy J. Halloran	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Carnegie Mellon University	**CI04-134**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
The Department of the Air Force AFIT/CIA, Bldg 125 2950 P St. Wright-Patterson Air Force Base, OH 45433	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**UU**	**33**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

In the mid-1980s the AFWC was formed, in-part, to transition the PME wargaming exercises from simulations hosted on remote Honeywell mainframes located across the city at Gunter Air Force Base. They were accessed via dial-up teletype terminals to a large state-of-the-art hardware–software wargaming system dubbed the Command Readiness Exercise System (CRES). The legacy simulations consisted of Honeywell FORTRAN programs used for various wargaming exercises (e.g., tactical, theater, and strategic) with primitive user interfaces (e.g., batch input and line-printer reports). CRES was designed to replace the Honeywell with in-house computer facilities. CRES used a Control Data Corporation (CDC) Cyber mainframe located in the AFWC building connected via serial lines to roughly 100 IBM PCs that were spread out among the game rooms and development areas within the building. The Cyber mainframe hosted a large FORTRAN based "Wargaming Engine." The PCs hosted player interfaces and acted as smart terminals to the Cyber. CRES was to be a general purpose wargaming system and was designed to support several wargames of various types. The AFWC building itself was designed and built to support CRES. It is a 56,000 square foot facility containing 22 seminar or game rooms and a "white" room to host game controllers as well as computer room facilities and office space for staff.

By 1988, for many reasons, the software development of CRES by a large government contractor had encountered problems. The contractor had put the bulk of their efforts into the design and implementation of the FORTRAN wargaming engine and done little to develop specific wargame instances usable by the PME schools—a critical capability from the Government point of view. However, contractor costs on the fixed-price contract were being overrun and the contractor was focused on finishing work by meeting only explicit contractual requirements. The AFWC leadership was also under increasing pressure from the PME schools to deliver a useable wargame exercise using the highly touted CRES. Against this backdrop the AFWC leadership decided to develop, using in-house Air Force, or "blue-suit" developers, a CRES nuclear wargame for the Air Command and Staff College (one of the four primary PME schools). I was one of the team involved in the development of the CRES nuclear wargame, later named *StratWar*.

This practicum examines the 18-month development of the StratWar wargame. The practicum is divided into two major sections. The first section presents a longitudinal narrative of the development of the wargame from its beginning to its first use by ACSC students. The narrative is presented by breaking the project up into rough temporal phases (corresponding, in the spirit of StratWar, to the flight phases of an ICBM) as follows:

- *Launch*—the genesis of the project. This section describes the operational concept of StratWar, introduces the development team, and points out up-front constraints (schedule, manpower, and technology) on the effort.

- *Boost phase*—the shaping of the wargame development from an idea to a tractable concrete project. This section describes the major efforts by the development team in the first 6–9 months of the project: the tedious reverse engineering effort to better understand the wargaming engine and the planning and design effort toward building a new user experience for the wargame.

- *Midcourse phase*—the StratWar development team, now a synergistic team making steady progress, is faced with significant technical and political problems as the StratWar software begins to take shape. First, two technical problems are described: creation and testing of the automatic targeter and the orchestration of the wargaming engine during the execution phase. Second, two serious political problems are described: (1) senior management's insistence for color player input forms and (2) technical leadership's insistence for unplanned testing to ameliorate their concerns about the robustness of the unorthodox network distributed design of the StratWar software.

- *Terminal phase*—finishing up development and hosting the first successful StratWar exercise for ACSC. This section discusses the last minute problems encountered right before and during the first StratWar exercise. These include: player training, dealing with 2167A documentation, and some graphical user interface design flaws.

The second section of this practicum undertakes a critical analysis of the StratWar software development building upon the issues described in the narrative. The analysis is organized around "what went right" and "what went wrong" during the project and proposes ten enduring lessons for software projects. A summary of "what went right" includes that StratWar

- worked and was a success;
- had, overall, throughout its development stable user requirements;
- had an end goal, the ACSC exercise, to guide it;
- had an effective development team
  - that confronted project risks and
  - made effective use of off-the-shelf software; and
- had no political "enemies."

A summary of "what went wrong" includes that StratWar

- didn't understand senior management needs;
- trusted, for StratWar, the contractor architecture of CRES; and
- purchased all the Sun 386i workstations before the project began.

The ten enduring lessons extracted from the analysis of "what went right" and "what went wrong" with the StratWar development project, and informed by the author's subsequent experience, are

1. Even successful projects encounter serious crises. Don't give up at the first sign of trouble—take responsibility and work toward solutions.
2. Reasonably stable requirements can mitigate the risk of a tight software development schedule.
3. Visible goals can help focus the team on the whole rather than the parts and avoid the serious problem of sub-optimization.
4. Effective teams take time to coalesce. Conflict is a normal part of the teambuilding process. Software practitioners must be open and motivated to learn new knowledge (e.g., technical or domain) all the time.
5. Mitigate risks early in the project—avoid the natural tendency to focus on "comfortable" (i.e., low-risk) work early in the project.
6. Make sure off-the-shelf software choices are deliberate decisions—not technical accidents.
7. Realize political "enemies" to your project may exist—keep senior management informed about problems and avoid surprises.
8. Work to elicit critical senior management requirements—realize what is important to you may be very different than what is important to them. Use Project Management Reviews, or a similar technique, to inform and stay in contact with senior management.
9. Understand, document, and analyze your software's architecture.
10. Purchase off-the-shelf computer hardware as late as possible in a project to lower risk and exploit Moore's law—both in terms of invested capital and hardware capability.

It is important to note that this practicum only describes a single experience. Hence, the validity of its lessons is suspect—to a degree. In addition, StratWar was a Government software development effort and may not be similar to all commercial software development efforts. For example, the StratWar development team did not have to deal with the dynamic of another company developing a competing product. However, it may be considered

similar to an in-house software development effort undertaken within a commercial company. Finally, the size of the StratWar software is modest—between half to three-quarters of a million source lines of code. About half of this code was newly developed during the StratWar project. The remaining half consisted of the wargaming engine code previously developed by the CRES contractor.

The primary and, I believe, enduring lesson of this practicum is to give the reader a sense of the troubles and even chaos encountered during a real-world software development project—even a successful one like StratWar. Armed with a decade of further software development experience I'm confident experienced practitioners will note similarities to personal experiences. However, if you find yourself asking: "What was wrong with these people? Why didn't they hire better people? Why didn't they use software process *X*, method *Y*, or tool *Z*?" you are in danger of missing the point: there is never a substitute for judgment, teamwork, individual hard work, talent, and sometimes luck. In my experience with StratWar and every software development project I have been involved with since, not all of them successful, every bit of knowledge about computer science and software engineering I knew has been put to the test—if you don't know it you can't apply it—and driven me to learn more.

Narrative

This section describes StratWar development from its genesis in fall of 1988 to the first operational use of the game by ACSC players in spring of 1990.

Launch

In fall of 1988 I was assigned to the StratWar development team:

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
CENTER FOR AEROSPACE DOCTRINE, RESEARCH, AND EDUCATION
MAXWELL AIR FORCE BASE, AL 36112-5532

7 NOV 1988

REPLY TO ATTN OF: WG

SUBJECT: CRES Nuclear Exercise Development Team

TO: WGSS (2Lt Tim Halloran)

1. I've selected you to be a member of the CRES Nuclear Exercise development team. This team will implement the game design previously agreed to by Air Command and Staff College (ACSC) and will be responsible for all activities leading to the execution of the wargame. Lt Col Burns is the team chief and overall project director. Major Kwalic is the ACSC representative to the team.

2. You will be assigned specific tasks for this project by Lt Col Burns through his primary assistants Major Lee, WGTA, and Capt Johnston, WGSS. These tasks include:

a. Developing, implementing, and documenting the new Computer Program Configuration Items (CPCIs) of Workstation Toolkit Subsystem (WTS) and Game Driver Subsystem (GDS).

b. Integrating the new Sun 386i workstation hardware and software into the CRES architecture.

c. Testing of the Wargame Subsystem CPCI leading to identification of problems, prioritization of proposed code changes, and code modifications.

d. Identifying database requirements, locating the appropriate data, and incorporating that data into the Cres Nuclear Exercise.

e. Participating in the exercise testing, preparation, execution, and post-exercise analysis phases.

3. The time available to develop and prepare this exercise is extremely limited; therefore, longer hours and the possibility of weekend work may be necessary. I expect your best efforts and full support to complete this exercise development.

GARY R. WARE, Colonel, USAF
Director
Air Force Wargaming Center

Strength Through Knowledge

As the above letter alludes, the StratWar project began with some constraints on its operational concept, development team, and technology.

Operational concept

StratWar was conceived to be a four day exercise where two teams of ACSC students develop and simulate execution of a Single Integrated Operational Plan (SIOP).

The SIOP is the overall plan that controls the US strategic (i.e., nuclear) missile, bomber, and submarine forces. It defines options for the President to execute in crisis. In StratWar, a completed SIOP contains ten execution options—each option is a self-contained pre-planned course of action usually increasing in severity from 1 to 10. For example, option-1 may disperse the bomber fleet to remote bases (to reduce vulnerability) and raise the alert level of the strategic missile and submarine forces, while option-10 might launch all nuclear missiles and commit all bombers to their assigned targets. The primary purpose of StratWar was to expose mid-career officers to some of the issues and complexity of creating a SIOP.

One group of students, called the "blue" player team, played as the United States while a second group of students, called the "red" player team, played as the USSR. A third group, composed of AFWC staff assisted by the ACSC faculty, acted as a "white" control team which monitored and guided the exercise. During the first three days of the wargame, called the *planning phase*, players would create, based upon a fictional world scenario developed for the exercise, a SIOP. During the fourth and final day, called the *execution phase*, the two sides would be placed in the role of national authorities in a highly tense world situation and be allowed to execute the planned SIOP.

Team

The development team was hand-selected by the center director and led by a project manager, Lt Col Burns. The StratWar project used a matrix organization with team members drawn from the various major functional branches within the AFWC.

- *Operators*. Air Force officers with real-world operational experience in the areas simulated by any wargame. This group ensures the validity and credibility of the wargame. In the case of StratWar, the operators all had either served as missile launch officers or bomber crews and some had staff experience involving the national SIOP. They interacted with the ACSC faculty representatives to ensure the wargame would meet the school's educational objectives. The project manager, Lt Col Burns was drawn from this group which, for StratWar, also included Lt Col Lee and Lt Col Bryant, and Major Wolczek.

- *Intelligence officers.* Specialists on the world situation and how information on the world situation is collected and distributed. For StratWar, this group consisted of Capt Catoe and Capt Witte.

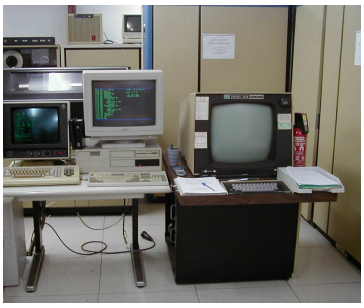
  The operators and intelligence officers worked together to develop the game databases which were carefully documented such that all data was drawn from unclassified sources. In addition, they worked with the programmers on the game user experience. Finally they wrote the StratWar player handbook, a user manual given to ACSC students.

- *Programmers.* The group responsible for development, testing, documentation, and maintenance of the StratWar software. In addition to the author, Lt Halloran, this group included Capt Johnston, Capt Kross, Lt Hutchins, Lt Meissner, SSgt Givens, and SSgt Adams.

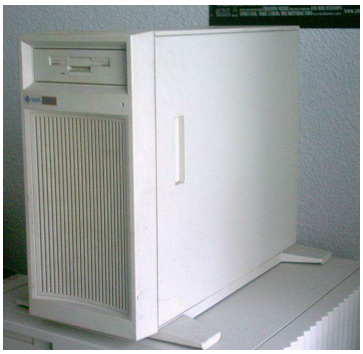
- *System administrators.* The group responsible for operation and administration of the AFWC computer systems. The primary system administrators involved with StratWar were Capt Atwell and Sgt Branson for the Sun computers and Mr. Pitts and Mr. Davis for the Cyber mainframes.

Technology

Both hardware and software technology constraints were placed upon the StratWar software development. The hardware for the project consisted of a development and production Cyber mainframe, player IBM XT PCs, and player Sun 386i workstations.



Consoles in front of a Cyber 180 mainframe



A Sun 386i workstation

The Cyber mainframes and the IBM PCs were already installed at the AFWC at the start of StratWar development—they had been procured and used by the CRES software contractor. The Cyber mainframe was the platform the wargaming engine ran on. The IBM PCs functioned as (slightly) smart terminal interfaces to the mainframe and were connected to it by serial cables. The IBM PC hosted the contractor developed Game Driver Subsystem (GDS) and the Workstation Toolkit Subsystem (WTS). GDS was the user interface on the PC for a wargame, while WTS was a communications library which helped GDS talk to the mainframe—both of these subsystems, while they worked, were perceived by the Government as failures that needed to be redeveloped from scratch—a richer, more interactive, user experience than they could provide was desired for StratWar.

The Sun 386i workstations were new to the AFWC. Prior to the start of StratWar development, the AFWC leadership had decided to purchase enough Sun 386i workstations to augment or replace the IBM PCs. This would put a single Sun 386i in every "red" and "blue" game room. (The Sun

386i workstation consisted of a 25-Mhz Intel 386 CPU, 4MB RAM, a 360MB hard disk, an Ethernet adapter, and a 15" monitor capable of 1152x768 pixel resolution at 256 colors.) AFWC leadership had been highly impressed with the state-of-the-art graphics capabilities of the 386i, a capability lacking on the IBM PC, and the capability to "emulate" MS-DOS, to support a smooth transition from the IBM PCs. Using both graphics and MS-DOS emulation together proved unworkable, however, because the Sun 386i, which ran a variant of BSD UNIX called SunOS, only emulated MS-DOS in a small 320x240 pixel window within the SunView windowing system (Sun wanted the 386i to serve as a "bridge" that allowed MS-DOS developers to port their applications to UNIX). To access the full graphics capabilities of the Sun 386i one had to work in UNIX not MS-DOS. Consequently, StratWar was developed under UNIX not MS-DOS.

The primary software constraint was that the wargaming engine was required to be used for simulation during the execution phase. It was understood by the programming team that the change to the Sun 386i would require new development of the entire user interface for StratWar.

Launch wrap-up

At project launch, team motivation was high—the challenge and importance of being directly involved with the creation of the first operational CRES wargame was exciting. The reality of the scale of this effort, hinted at in the tasking letter's notes about "longer hours" and "weekend work," would soon become apparent to the team as the project got underway.

Boost phase

The early StratWar development work comprised two distinct engineering efforts: understanding how the wargaming engine worked and planning the new software development on the Sun workstations.

Understanding the wargaming engine: the "operator cave"

The operators and intelligence officers along with Lt Meissner and SSgt Givens were faced with the task of making the contractor-provided wargaming engine usable for the StratWar game. Although documentation for the wargaming engine existed, the documentation was closer to a detailed reference manual on input syntax rather than a description of how the simulation behaved (i.e., syntax-focused rather than semantics-focused). In addition, the team had very limited interaction with the contractor who had created the wargaming engine. Hence, understanding strategic nuclear simulation in the wargaming engine became a reverse engineering effort that would go on for roughly 8 months.

The wargaming engine accepted input only in the form of an orders file. This file contained a series of orders, in plain text, for the simulation. These

included both simulation data and directives. Simulation data included information such as base locations and assignment of aircraft to bases. Simulation directives included tasks such as when (in simulation time) to output reports, read another orders file, output a "checkpoint" (a saved game that could be restarted later), and terminate the simulation. Once an operator had created an orders file, running the wargaming engine was straightforward. A batch job was submitted to the Cyber to run the wargaming engine which would simulate until the termination directive was reached. Reports, representing simulation results, would appear in files and could be examined by the operators. Based upon examination of these results the orders file could be modified and the entire process repeated…over and over again.

This effort was tedious and the frustration levels often ran high early in the effort. The team lived in a small room, a sort-of "operator cave," filled with Cyber mainframe terminals and wall papered with mainframe printouts of orders files and FORTRAN code, maps of the US and USSR, and to-do lists. Complaining, early on, about the exacting orders syntax and the brittleness of the wargame engine itself reached such a level that the team self-imposed a "complaint tax"—no complaining was allowed about the wargame engine unless you put a quarter into a large glass jar located on a cluttered table in the middle of the room.

Passersby during this time would peer into the cave door's little window in wonder at the busy group working in this cluttered room filled with maps and line printer paper on every wall and draping every table. Most of the group would be hovering over amber mainframe terminals from which someone would jump up, toss money in a jar, and then rant loudly for a minute or two before sitting right back down at the same terminal to continue with barely a glance from the others.

Planning the development of a the StratWar user experience

The second team, composed primarily of programmers but guided by Lt Col Burns, had a much different task. They would create the entire user experience for StratWar using the Sun 386i workstations and develop control software to drive and communicate with the mainframe for the execution phase of the wargame.

The most severe risk for this group was soon apparent: technical knowledge. None of the team, except Capt Atwell, had ever used—let alone developed software for—a UNIX workstation like the Sun 386i before. Capt Atwell had encountered one briefly in a class he had taken at the University of Utah. Thus, Capt Atwell immediately became the de facto system administrator. Capt Kross had recently graduated from the Air Force Institute of Technology where he had worked on the use of a relational database management system (RDBMS) for wargaming. Lt Halloran and Lt Hutchens

were recent graduates of the computer science program at the Air Force Academy and came to the project with Pascal and assembly language experience. Capt Johnston focused on management of the project and would not be involved with the day to day software development. The core programming team had a solid foundational knowledge of computer science but lacked knowledge and experience with the specific hardware and software technologies being used for the StratWar project. Despite this problem, the team faced a large software development effort with a hard 18-month deadline. The approach taken to mitigate this risk was two-fold: (1) make heavy use of off-the-shelf software and (2) select a software design that would minimize the amount of new technologies each team member would be required to learn.

Off-the-shelf software

The selection and use of off-the-shelf software hinged upon the judgment of two team members: Capt Atwell for the X Window System and Capt Kross for the Oracle RDBMS.

The smooth introduction of X

In 1988 Sun workstations provided a graphical windowing system, a mouse, and built in Ethernet. Sun, like all workstation vendors at that time (e.g., Apollo, Digital, HP, and Tektronix) provided a proprietary windowing system with proprietary APIs—for Sun it was SunView. SunView seemed the obvious windowing environment to program the graphical map situation display for the StratWar execution phase. However, Capt Atwell advised that we use the X Window System instead. X was a research project developed at MIT (using ideas taken from the Stanford W windowing system—hence the name). Capt Atwell believed that X would replace all proprietary UNIX graphical windowing systems and provide a much needed way to move graphical applications from one vendor's workstation to another. In addition, X allowed displaying a graphical application remotely over a network (i.e., running an application on one machine (like a mainframe) but displaying its user interface on another (like a Sun), much like telnet already provided for command line UNIX users).

Without fanfare, X was selected by the development team for use over SunView and Capt Atwell obtained the source code for X11R4 from MIT and compiled it for the Sun 386i. Savvy use of X would solve several difficult technical problems encountered during StratWar development and having its source code would, in fact, save the project from a later political crisis. The introduction of the Oracle RDBMS would not go so smoothly.

The rocky introduction of Oracle

Data management for all the legacy wargames was file-based. The idea of using a RDBMS had been researched at AFIT but had never been used for a production wargame. The AFWC, at this time, sponsored several AFIT

simulation and wargaming research projects being conducted at Wright-Patterson Air Force Base in Ohio. Capt Kross, a recent AFIT graduate and member of one of the AFIT/AFWC research projects, advocated use of an RDBMS to help mitigate the risk imposed by the 18-month development schedule. Capt Kross was especially interested in the so-called "4GL" tools provided with commercial database products. These tools allowed creation of terminal-based text input forms and reports from database tables. In addition, the flexibility of the RDBMS would allow rapid change of the data schema as the operators gained understanding of what information the wargaming engine required. AFIT had conducted its research projects using the *Ingres* database product; however a new product, called *Oracle*, was of interest to Capt Kross due to a reputation for superior tool support. Both products were brought in and evaluated by Capt Kross and Lt Hutchins and Oracle was the clear winner—the Oracle tools, SQL*Forms and SQL*Reportwriter, were superior to the Ingres offerings at that time and the core RDBMS was adequate in terms of performance. Cries of "Ingres sucks; Oracle rocks!" were often heard in the cubicles during this tool-focused evaluation.

Despite the outcome of the Kross–Hutchins evaluation, monetary and political obstacles were raised against using Oracle for StratWar development. First, Oracle was expensive to license. Second, a historical bias towards use of file-based systems existed among senior AFWC technical leadership. Third, despite supporting RDBMS use for a production wargame, the AFIT research teams strongly preferred the AFWC adopt Ingres rather than Oracle.

Lt Col Ellertson, the functional head of the programmers, system administrators, and computer operators at the AFWC, had a bias toward use of file-based systems—they were proven, fast, conceptually simple, and free. The RDBMS was perceived as slow, complex, and costly. However, Capt Kross was able to make a convincing case, backed up by research experience at AFIT and the in-house evaluation, that flexibility would be increased and development time would be reduced, albeit with the trade-off of licensing costs, through the use of an RDBMS. Thus, Lt Col Ellertson agreed to support using an RDBMS for StratWar. However, both he and AFIT would prefer StratWar use Ingres. From Lt Col Ellertson's point of view, the AFWC already had contracts in place for Ingres due to its use on wargaming research at AFIT. Augmenting an existing Government contract for Ingres was a simple task compared to engaging the bureaucracy to create a new one for Oracle. The AFIT researchers, unconcerned about contracting issues, argued the superiority of Ingres based upon capabilities of the core database management system. Ingres, at that time, supported far more of the relational capabilities deemed significant by researchers than Oracle.

Oracle, despite all these obstacles, did win out over Ingres. Capt Kross was able to convince Lt Col Ellertson, an experienced mainframe wargame programmer, that the surrounding tools and not the database server were the largest benefits of the RDBMS for StratWar—primarily to mitigate schedule risk. In addition, Government contracting officers were able to negotiate a reasonable price for Oracle software licenses. Hence, Oracle was licensed and installed for use on the Sun 386i, where it would become the backbone of all new StratWar development, and the Cyber mainframe, where it would languish unused.

Constraining the software design

The overall software architecture and design for StratWar evolved during development but followed two key constraints identified by the programmers early on: limiting use of the ill-understood wargaming engine to only the execution phase and limiting the amount of new technology each programmer had to learn to be productive.

Early on, the programming team noticed that the wargaming engine was only required to simulate the execution phase of StratWar. Hence, a design decision was made to develop and execute the entire planning phase of StratWar on the Sun workstations. The planning phase of the game would use a repository-based architecture. All planning data would be stored in the Oracle RDBMS and the players would interact via forms and reports built using SQL*Forms and SQL*Reportwriter. Hence, for the first three days of the game, red and blue players would interact with the game using a menu of text based forms run on the Sun workstation in their seminar room.

As the software development effort began to take shape, the list of training required by programmers started to grow long and daunting: UNIX, X graphics programming, C language programming, SQL, Oracle tools (SQL*Forms, SQL*Reportwriter), serial communications to the Cyber, and so on. Unfortunately, there was very limited time and funding to send programmers to formal training—the team would have to learn almost all the unfamiliar technologies on-the-job. To mitigate this risk the design was partitioned such that no programmer would have to learn all the technology. Roughly, the division was drawn between Oracle tools, X graphics programming, and any work on the wargaming engine. Capt Kross and Lt Hutchins (later joined by TSgt Adams) would work all Oracle development, Lt Halloran would work all X development, and Lt Meissner and SSgt Givens would work any required changes to the wargaming engine. Some overlap occurred as the project progressed but, in general, this scheme proved workable and was successful.

One further division emerged which would benefit the project. That was a clear separation of system administration of the Sun workstations from the

software development effort. Lt Hutchins, who fancied himself a "hacker" in the original sense of the word, advocated that all the Sun programmers should have carte blanche *root* (administrative) access to the Sun computers. Capt Atwell resisted and was able to quickly setup a reasonable stratum of tasks that required him and tasks that programmers could perform. In practice, this simplified learning for both groups—the programmers focused on learning how to use UNIX while the administrators focused on learning how to administer UNIX.

Boost phase wrap-up

The beginning of technical work on StratWar was a difficult period—filled with long hours and frustration. The operators were holed up in the "operator cave" and the programmers were busy trying to figure out, with sketchy but growing technical knowledge, how to structure and carryout the software development efforts. The decisions described above were often only reached after long arguments between team members with starkly differing ideas about what the best solution was. To an outsider the project might have appeared doomed—a complaint jar full of money but no working strategic simulation, a group of programmers arguing for weeks over what data management system to use when they could hardly edit a text file on their computers, system administrators fumbling around with coax cable and transceivers draped around and over office cubicles trying to get the new "PCs" to communicate when mainframe terminals always worked properly. However, appearances can be misleading, and in this case the formation of a solid team was underway, risks were being tackled—not being ignored—and progress was being made toward understanding how to construct the complete StratWar exercise for ACSC.

During this period the overall design for StratWar came together. It had transitioned from an idea to a concrete plan. The operational concept of the StratWar game, developed by Lt Col Burns and the ACSC faculty representatives, remained stable. This was critical to the technical team as they could work to solve the issues described above (reverse engineering, off-the shelf software selection, and technical training) without major shifts to the core user requirements.

Midcourse phase

By roughly 10 months into the StratWar development project a nascent wargame had been created. The Oracle forms and reports were well underway for the planning phase of the wargame. SSgt Adams had joined Capt Kross and Lt Hutchens to assist with development of these forms (roughly 60 were required). The operators had uncovered most of the wargaming engine functionality needed for the execution phase of StratWar and subsequently the Oracle database schema on the Sun workstations was experiencing less change. Lt Halloran had developed a graphical map

situation display for the execution phase that tracked missile, bomber, and submarine activity and allowed player interaction. Requirements feedback from the operator team and the ACSC faculty representatives about the emerging wargame was ongoing and positive.

Progress at this point in the project was steady—a strong development team had emerged and completion of the project, on time, seemed possible. However, during this time period (from roughly the tenth to the sixteenth month of the 18-month project) several serious technical and political problems emerged that could have made the project fail. This section will describe four of the most serious of these problems, two technical and two political, and how they were solved—some by hard work, some by creativity, and some by luck.

Technical troubles

The StratWar software design had many technical risks: some were caused by the evolving technical knowledge of the programming team, while others were due to the tight schedule of the project. Most of these risks did not materialize into real problems and many that did were straightforward for the team to deal with. A few caused serious problems that had to be dealt with for the project to succeed. All of these involved communication between the "front-end," the term used for the collection of all the software hosted on the Sun workstations, and the wargaming engine. Two that caused serious problems were game data duplication and orchestrating the execution phase.

The automatic targeter and game data duplication

The transition from the planning phase to the execution phase required that the student's SIOP plan be output into a form the wargaming engine could simulate. This was troublesome because the wargaming engine simulated at a very low-level (e.g., base bomber-1 and bomber-5 at base-1, fly bomber-10 to location *x*, launch ICBM-50 at location *y*, etc.) while the student SIOP planning was more aggregate (base 50 bombers in this region, this target has a higher priority than this one, allocate 5 ICBMs at this target priority group, etc.). To translate the student SIOP into a form the wargaming engine could simulate, a program called the "automatic targeter" was conceived. This program was a large Oracle program written in Pro*C (a C program with SQL statements embedded in it) by Capt Kross and Lt Hutchens. This program encountered several problems.

First, the use of Pro*C proved troublesome. The Oracle team, unlike the X team, wrote very little C code and had difficulties with the tricky portions of the language. In addition, the Pro*C program was pre-compiled into C source code—this made program crashes difficult to debug due to the rewriting of the program source code by the pre-compiler. Second, the program was slow. The final program took 2 to 3 hours to run during the first StratWar game. Hence, testing at scale was tedious. Finally, and most

seriously, data inconsistencies between the front-end and the wargaming engine were generally discovered by this program.

The operator team that developed the StratWar exercise data was the same team who in the "operator cave" used orders files to figure out how the wargaming engine simulated strategic nuclear war. Due to this group's comfort with the orders file format and the aggregate nature of front-end SIOP planning, the technical team decided that only the minimum world situation data required for planning would be loaded into Oracle. The full world situation data would exist in a master orders file for the wargaming engine. The subset in Oracle would be derived from this master. After the automatic targeter was executed, the resulting orders file fragment would be combined with the master orders file to produce a full StratWar orders file suitable for simulation by the wargaming engine.

The problem with this scheme was that if you ran the automatic targeter and the wargaming engine crashed while executing its output, you were unsure if you had a data problem or a code problem. This was a tremendous architectural mistake: consequences included a schedule cost to the project and aggravation to the team. The programmers and the operators had to maintain consistency between the master orders file and the Oracle subset of this data at a time in the project when the programmers badly needed the data to stay the same so they could test software and the operators badly need the data to change so they could meet the operational requirements of the wargame. This mistake was never solved for StratWar—later CRES games would require that all scenario data be managed in Oracle so that it was easy to maintain several different scenario sets for software testing independent of the ongoing exercise data development.

Orchestrating the execution phase

The execution phase, the fourth and last day, of the wargame required carefully orchestrating nearly all the software developed for StratWar. The wargame engine would simulate, in roughly twice real-time, the unfolding of a simulated nuclear war using the "red" and "blue" SIOPs developed in the planning phase of the game. Each "red" and "blue" player room would have a polar situation map, developed by Lt Halloran using X, running on the Sun workstation which would be updated every 2 minutes. The players used one of the IBM PCs (acting as a vt100 terminal connected to the Sun workstation) to input orders and send email to the opposing side. Game controllers, called the "white" team, composed of AFWC operators assisted by ACSC faculty needed to be able to start, pause (for academic discussion), speed up, or slow down the simulation at any time. They also, unknown to the students, could create events that might raise tension between the opposing sides (e.g., make phantom missiles or bombers appear on one side's map display). Eleven games would be played at the same time using 22 player Suns and PCs with

all 11 wargaming engines running on a single Cyber mainframe. Orchestration of execution day eventually worked but the road traveled was very rocky.

The risks involved with the execution phase were not fully confronted early in the StratWar development. This was due, in part, to a trust in legacy capabilities. The CRES contractor had developed a toolkit capability to communicate with and control the wargaming engine on the Cyber mainframe from the IBM PCs. The original technical plan, visible even in the tasking letter from leadership to the development team, was to "clone" this capability on the Sun workstation. This part of the project was the Workstation Toolkit Subsystem (WTS). Lt Hutchins took on this task as a side-project and to learn the C programming language. WTS programming pre-dated work on the automatic targeter discussed above. WTS was plagued with problems from its beginning. Lt Hutchins rarely had enough time to devote to it, and the serial connection to the Cyber from the Sun was prone to intermittent failure, requiring a reset at the mainframe terminal controller to fix it, so progress was slow and tedious. In addition, WTS was the only time we lost the source code for a program. Lt Hutchins had added to the WTS program the ability to upload a file from the Sun to the Cyber using the x-modem protocol. This ability would be used to place an order file fragment (e.g., disperse my bombers, execute SIOP option 2) onto the Cyber for the wargaming engine to read. WTS would upload the file to the Cyber and then delete it. Unfortunately, the file used for the first test was the C source code for WTS and only half the program worked—the copy to the Cyber failed but the source code was deleted from the Sun disk. Capt Atwell was able to recover the source, but only an older version, and roughly 2 weeks of work had been lost and the overall WTS development effort would never recover.

A second serious risk was the capacity of the Cyber mainframe. Execution day required 11 wargaming engines to be running simultaneously on a single Cyber for roughly 4 hours. The 11 simulations needed to be able to simulate at a rate roughly twice wall clock time (i.e., 2 minutes of simulation passed for each minute of wall clock time). Early on, the operators working with the wargaming engine in the "operator cave" expressed concerns that this might not be possible—they were having difficulty getting 3 or 4 wargaming engines to run simultaneously at a game time to wall clock time ratio of 1 to 1.

Faced with unreliable WTS software and a Cyber mainframe not capable of running 11 simulations at a reasonable rate, the project clearly had a large problem—one that could have caused the entire project to fail. In this case, however, the StratWar development got lucky—both problems disappeared because the mainframe and its serial connections did as well.

A separate AFWC project had ported a legacy Honeywell FORTRAN wargame to the Cyber and a PME school wargame had been hosted at the AFWC using this software. Although the overall exercise was successful, at least from the student point of view, the automated portions of the exercise had been a disaster. In a nutshell, serious capacity and communication problems had occurred while using the Cyber version of this wargame. After a careful investigation and analysis of the problems, AFWC leadership decided to replace the Cyber mainframes with newer machines that had, roughly, six times the capacity of our current ones, which had been around since the contractor started development of the CRES wargaming engine. Far more important to the StratWar development was the fact that the new mainframes used TCP/IP Ethernet communications rather than serial lines: the mainframe could now communicate with the Sun workstations via the network rather than via a serial cable.

We now had a Cyber capable of simulating 11 wargaming engines for StratWar faster than the desired speeds and talking to the Sun workstations via the network. However, orchestrating the execution phase was still an unsolved problem. The addition of the mainframe network capability meant the WTS had to be thrown out—a completely new design was required. Lt Hutchins was really busy with the Oracle front-end work and work on the automatic targeter loomed. Because the map situation display was nearly completed, Lt Halloran took over the effort.

The X Window System enabled real-time control of the execution of the wargaming engine. The new Cyber included a client-side implementation of the X protocol. Lt Halloran wrote 500 lines of C code that linked into a FORTAN interface developed by Lt Meissner. When the wargaming engine was started, it now displayed a small clock window on a Sun workstation—taking advantage of the ability provided by X to separate the client program from the display. This small clock window displayed the current simulation time of the wargaming engine. The clock window also registered several memory areas in the X server that allowed another X program to control the execution of the wargaming engine. This second program, used by the "white" controller team, provided a graphical user interface for wargame engine control (e.g., stop, start, pause, set or change the ratio of wall clock time to simulation time).

To transfer data back and forth between the Cyber and the Suns, NFS was the obvious choice. The Network File System (NFS) developed by Sun was a de-facto industry standard that allowed disks on remote computers to appear local. Once the new Cyber was installed, using NFS was attempted for StratWar by Lt Halloran—it failed. The Cyber implementation of NFS only allowed the mainframe to act as the NFS server (i.e., you couldn't mount Sun disks to make them visible to the mainframe). Hence, the approach was to

have each Sun workstation "mount" (i.e., attach) to a disk on the mainframe. This approach didn't work. The problem was that the Cyber NFS server implementation maintained no consistency between the disk state seen via NFS and the disk state seen directly on the mainframe. Wargaming engine orders files copied from the Sun via NFS just didn't exist when the wargaming engine tried to read them. CDC claimed our use of NFS was the problem because, from their point of view, NFS only allowed "lesser" machines, like our Sun workstations, to use reliable storage on the mainframe. After a good deal of argument CDC did provide us with a version of NFS that maintained consistency—at the cost of using over 80% of our mainframe CPU capacity toward this purpose. Lt Halloran had to give up on using NFS due to these problems with the CDC off-the-shelf software. The final solution for the wargame was to use FTP (file transfer protocol). FTP was a well established tool that encountered no consistency problems on the mainframe and had a very low resource cost on both the mainframe and the Sun. However, FTP required more programming work as it operated at a much lower level of abstraction than NFS. Lt Halloran, assisted by the Cyber and Sun system administrators, had to create a slew of special purpose scripts to drive FTP communication between the Sun and the Cyber.

Hence, albeit later than expected, orchestration of the wargame engine was successfully engineered into StratWar. It was now possible to support the execution phase of the wargame. At this point, demonstrations of the developing wargame became far more dramatic and interesting because the execution phase could be shown. SIOP options could be launched and the effects seen on the Sun map graphics situation display—sometimes thousands of ICBM launches at a time from both sides.

As difficult as the above technical problems were to solve, none of them at any time reached the level of crisis that the problems described below did. These were political problems and, at least the first, would bring the project to the eve of being cancelled.

Political troubles

In addition to technical problems, serious political problems were encountered during this phase of the wargame software development. As any introductory textbook on project management will state: the success of any project is dependent upon support from the senior management of the organization [14]. A lack of sensitivity to the political needs of AFWC senior management created two problems the StratWar team was forced to solve.

Saving the project with a color X terminal

Roughly a year into development, the "operator cave" had been converted into a test room for the emerging StratWar software. The operators had, in a sense, tamed the wargame engine and were now focusing their efforts on guiding the new user experience and beginning to develop the materials (i.e.,

player training and documentation) which would augment the software to make up the complete StratWar exercise. During this time, senior management started being shown the planning phase software by the operators—who, like the programmers, were eager to show off the emerging wargame. Senior management was impressed with the game but wondered "why are the player forms only using two colors?"

The player input forms, developed using the Oracle SQL*Forms tool, ran within the X terminal program on the Sun 386i (i.e., a terminal window). Even though the Sun workstations supported sophisticated color graphics, the X terminal text was black and white—actually, any two colors could be selected, but only two. Senior management had purchased the Sun 386i workstations because they had stunning color graphics and they would not allow the first 3 days of the StratWar wargame to use black and white player input forms. This quickly became a gigantic problem for the development effort with a series of heated meetings arguing the positions: "Just add color to the player forms," and "We can't. Oracle*Forms can't do that—it's just not possible!" This reached the point where the project was going to be cancelled and "reset" to use technologies that could produce better colors.

Finally, when it was clear deadlock had been reached, a meeting was planned to stop and re-plan the project. However, on the eve of that meeting several of the programmers were explaining this situation to Capt Atwell and the other system administrators who were confused about what was going on and how the project could be killed due to a lack of color screens. As the description of the problem was narrated, Capt Atwell interrupted, "but you have the source code for the X terminal program don't you? Just add colors to it?" This was an insight that had not occurred to any programmer on the StratWar team—quickly a solution emerged: change the X terminal source code such that a "gel"-file could be given when it was started ("gel" as in the use of colored plastic gels to color theater lights). This gel-file would contain an 80x25 character block describing the color at each cursor position on the terminal screen. Create a gel-file for each and every SQL*Forms player input form and they would appear to be in color.

This solution worked and saved the project. Lt Halloran made the modifications to the standard X terminal program to create the color X terminal. Capt Johnston, who had focused on management until this time, crafted the gel-files. The Oracle development team was not affected.

Not using the mainframe and the Saturday testing crisis

As the project headed for completion, and, in part, due to the color X terminal crisis, Lt Col Ellertson became more interested in the robustness of the StratWar software. He also became aware that the entire planning phase of the wargame, three of its four days, did not use the Cyber mainframe at all.

Lt Col Ellertson was an experienced programmer but, like most senior Government computer professionals of this time, was much more comfortable with mainframe technology, such as the Cyber, than networked workstations, such as the Sun 386i. He became increasingly concerned that the reliance upon the Sun workstations would make the game unreliable and prone to crashing. After all, from his point of view, the Cyber filled a room, had on-site CDC maintenance and an army of computer operators to keep it running. In contrast, the Sun workstations were tiny, had slow phone tech support, were all over the building, and had only Capt Atwell to keep them going. In addition, the AFWC Sun 386i workstations had been delivered with a bad batch of hard disks: roughly 40% had failed and had to be replaced by Sun in the first 6 months after the workstations were installed.

A "Saturday Test" was scheduled to test the entire configuration of an actual StratWar exercise and to stress it by introducing failures. Lt Col Ellertson would have random Suns turned off and would observe how the software responded—did killing one Sun cause lots of player rooms to be impacted? Was each Sun a single point of failure for the entire wargame? The answer was no—the software passed this test with no serious problems. The real Achilles' heal of the Suns, the Ethernet network through which communications between the workstations passed, was never broken during the Saturday test (and would never fail during any real StratWar exercise). Lt Col Ellertson was convinced of the robustness of the StratWar design and began to understand that computers like the Sun workstations would take an expanding role in future wargaming software.

Midcourse phase wrap-up

Despite the technical and political problems that cropped up during the long middle portion of the StratWar development the team made consistent progress toward completing the project. There were far fewer long days and the whole development team worked together in a synergistic fashion.

By 16 months into the development, a complete StratWar wargame had emerged. The final two months would be filled with planning and preparations for the first real StratWar exercise. Team moral remained high, and in fact grew, as the date for the first exercise grew closer and the remaining development tasks were completed.

Terminal phase

The last two months of the StratWar development were exciting and busy. The team scrambled to get ready for the debut exercise with ACSC. More of the ACSC faculty were introduced to the wargame and after negotiation between AFWC and ACSC plans solidified around a 5-day exercise held during a single week in the spring of 1990:

- *Day 1: Training.* Monday would be hands-on training for the ACSC students. Before this day, student "player handbooks" (i.e., game manuals) would be distributed to ACSC students to prepare them for the time limited hands-on training. The player handbooks had been authored by the operator portion of the StratWar development team and comprised roughly 100 pages describing the general exercise context and scenario, user instructions for the "red" or "blue" player portions of the software, and a reference for all the data used in the game (e.g., weapon types, bases, etc.).

- *Day 2–4: SIOP Planning.* For three days, each team would, using the SIOP planning forms and reports, develop a StratWar SIOP. This represented the bulk of the exercise and was augmented by faculty guidance and strategic discussions—creating a StratWar SIOP required many trade-offs that were, by design, difficult for students to make. At the end of day 4 the planning information would be run through the automatic targeter by the "white" controllers.

- *Day 5: SIOP Execution.* Friday, the student teams would be placed in the role of national command authorities and played out a tense world situation. The students would monitor the strategic situation on the Sun workstation using the map situation display. A single PC, connected to the Sun workstation and acting as a text terminal, would allow a team to send email to their opposing team and execute options of their SIOP.

Due to the large number of ACSC students, two StratWar sessions would be needed: a morning session and an afternoon session. The morning session would have 11 "red" and 11 opposing "blue" teams, as would the afternoon session. Each team contained roughly 15 students and faculty.

The team dealt with several challenges during this phase of the project. The pressure was on for the StratWar developers to produce 2167A documentation. Player on-line training during day 1—a requirement missed by the programmers—needed to be supported. Finally, the actual StratWar exercise with ACSC had to be executed.

Documentation and document-driven development

As the ACSC exercise drew closer, but throughout much of the software development, the programmers were faced with increasing pressure from functional branches (specifically configuration management, testing, and computer operations) working for Lt Col Ellertson to produce 2167A software documents. DOD-STD-2167A "Defense System Software Development" [8] was the approach used for software development. DOD-STD-2167A directed a document-driven waterfall software development

methodology—the progress of a software project was measured by what documents had been completed not by how much of the software worked. This methodology was not faithfully followed by the StratWar project because of the good judgment of Lt Col Ellertson, who understood that extreme "by-the-book" interpretations of 2167A could cause a software development project more harm than good.

Extreme interpretations of 2167A included banning any code development until a full specification, design, and program pseudo-code had been written. As a concrete example, such an extreme approach had produced the wargaming engine. The contractor had, in fact, been mandated to produce one line of pseudo-code for each line of FORTRAN. In the era before structured control flow was common in commercial programming languages the pseudo-code represented the structured version of the program while its actual implementation would be some form of assembly language. By 1988, even within conservative Government software development efforts, use of higher level programming languages had made this type of situation a historical curio.

Cyber FORTRAN, the language used to develop the wargaming engine, supported structured control flow and the 1 line of pseudo-code to 1 line of source code mandate created a "double-vision" within the FORTRAN code as the below (contrived) example illustrates:

```
C LOOP MONTH FROM 1 TO 12
      DO 20 MONTH=1,12
C   SUM = SUM + DMG(MONTH)
          SUM = SUM + DMG(MONTH)
C END LOOP
20    CONTINUE
```

In addition, the contractor, who had created the FORTRAN code using a pseudo-code to FORTRAN pre-compiler, didn't want the pseudo-code removed from the wargaming engine source code (nor did they want to provide the pre-compiler to the Government). It would be a long time after the StratWar development was completed but eventually the pseudo-code was removed from the wargaming engine. This was a good decision because FORTRAN, as a standard language with commercial support and training available for it, was far easier to maintain than the contractor pseudo-code. The front-end development of StratWar, while required to use 2167A, would not suffer as the wargaming engine did from an extreme interpretation of the directive.

Due to the tight schedule and the technical judgment of Lt Col Ellertson the StratWar front-end development produced little more than outlines of the required 2167A documentation. The full set of documents for the StratWar

software were produced during a 4-month period after the first exercise was completed (over the summer of 1990). One problem the development team encountered during this effort was that 2167A proved ill-suited for software, such as StratWar, that made widespread use of off-the-shelf software. 2167A tacitly assumed that all the key components of the final software system were developmental items or that they were previously developed by a government project that had used 2167A. For example, how does one assure that the non-existent software requirements specification for Oracle's SQL*Forms maps to and is consistent with the StratWar system specification?

The team used an automated off-the-shelf document publishing system, called Interleaf, to create the StratWar technical and user documentation. Use of Interleaf was critical to the timely creation of StratWar documents—not because of its editing capabilities per se, but because of its de facto capability to manage configuration of documents being worked on by many people at the same time.

The final utility of the StratWar 2167A documents is dubious; however they did capture some useful design information including how all the software components were put together to run an exercise. Combined with the detailed and high quality user documentation created by the operators for the "player handbooks," StratWar ended up with a reasonable set of software documentation; albeit sometimes scattered across several documents due to its 2167A-derived organization.

Student player training

As the plan for training the ACSC students was discussed with the programmers a problem was discovered: no provisions in the software had been made to support on-line training for students. During the Monday training day the ACSC students would get to try out and become familiar with each part of the StratWar software: the SIOP planning forms, the graphical map situation display, etc. To support this required the software to support "hopping" between portions of the game—a problem because the information required for the each step of StratWar was created in the previous step. Capt Kross and Lt Hutchins worked with the operators to populate a "training" Oracle database that would be complete enough to allow students to bring up any planning form in any order they wished (and to support multiple students trying out each form). Because it took hours to execute, it was out of the question to invoke the automatic targeter during student training. Therefore a "training" wargaming engine orders-file was created that allowed a quick transition from hands-on use of the planning forms to live use of the map situation display coupled with a running wargaming engine simulation.

The StratWar 1990 exercise

The first 5-day StratWar wargame for 500 ACSC students occurred, as planned, in spring 1990. It was an overwhelming success and was later voted the "highlight of the academic year" by the ACSC students and faculty—a first for any AFWC wargame. The entire development team assisted during the execution. The operators acted as "white" team controllers. The programmers and system administrators acted as technical support staff.

Training and the entire SIOP planning phase went smoothly. The fourth night of the game ended up running very late for the programmers and game controllers, but not the ACSC students, because the automatic targeter had to be run on the 22 Oracle instances containing the SIOP planning database (11 for the morning session and 11 for the afternoon session). Over the next several hours, as each of the 22 automatic targeters finished, the programmers and operators reviewed the output—most were declared ready for execution day. However, mostly due to SIOP planning mistakes, a few had to have the student's SIOP plan altered by the game controllers and the automatic targeter restarted. Despite a very late night, all the SIOPs were ready for the simulation by the wargaming engine on Friday morning. The automatic targeter was invoked from the "white" control room but actually ran in a distributed fashion on the 22 Sun workstations located in the player "red" and "blue" seminar rooms—this created an eerie flickering surreal scene in the dark player seminar rooms as the disk light on each Sun quickly blinked on and off for several hours. This was strange enough to cause the AFWC security guards to inquire what was happening in the player seminar rooms.

Execution day, the most complex part of the StratWar exercise, went well. Throughout the entire day only a single game had to be stopped and restarted due to an unknown "freeze" of communication between the wargame engine and the map situation display software. The restart only took a few minutes and that game had no further problems.

Problems with the graphical user interfaces developed, primarily by Lt Halloran, became apparent during execution day. Two examples were the lack of an overview map and the lack of confirmation dialogs for critical control actions on the wargame control interfaces.

The map situation display allowed zooming of its view to see a close-up of an area of the map. The user interface problem encountered was: when the students were in a zoom they could no longer see the entire map. This caused at least one team to be wiped out when the other side launched all their missiles during a time when the team had the map zoomed to watch bombers flying (dispersing from a primary base to a remote base) in the northern part of the US. Their first indication of the attack was when their screens turned yellow (the map situation display painted yellow circles

indicating nuclear explosions)—when they zoomed out they found a mass of destruction had been inflicted by the opposing side. Future wargames would include a small overview map to keep situational awareness even when focused on a small geographic area.

The second user interface problem involved the graphical controls for the wargaming engine. The control screens didn't include confirmation dialogs for critical actions. During the execution day, one of the game controllers had the Sun screen saver start-up. To cancel the screensaver he clicked his mouse which just happed to be over the "terminate game" button of one of the four StratWar games he was controlling. That one game quickly shutdown and had to be restarted.

These user interface design flaws would probably not occur today. But it would not be until 1991 that Visual Basic would be released for Windows and bring graphical user interface development to a much wider practitioner audience. In addition, by the mid-90s, books such as [5] and [18] were, and still are, widely read by practitioners. Even in 1990 these problems could have been uncovered through usability testing, but testing of StratWar focused on functionality and the user interfaces had only ad hoc usability testing.

Terminal phase wrap-up

StratWar, while successful, would have a very short lifecycle. The second ACSC exercise would be held in spring of 1991—the last the author would be involved with. The wargame was in-use for a few more years but by then the changing world situation made it obsolete. The end of the cold war caused changes to the ACSC curriculum that ended the requirement for StratWar.

Analysis

Building on the issues described in the above narrative, this section undertakes a critical analysis of the project from the author's point of view. It has been over a decade since the project was completed but several of the lessons that can be learned from "what went right" and "what went wrong" on the StratWar project are timeless and can occur on any modern software development effort.

What went right

StratWar worked and was a success. This is easy to overlook but many software development efforts fail. In fact, as the narrative illustrates, StratWar had many opportunities where it could have failed. The political crisis of the color X terminal (had SunView been used instead of X and we didn't have the terminal emulator source code) or the technical difficulties of orchestrating the execution phase of the game (without having a new network-connected mainframe appear just in time) could have caused the overall effort to fail.

> Lesson: Even successful projects encounter serious crises. Don't give up at the first sign of trouble—take responsibility and work toward solutions.

StratWar had, overall, throughout its development stable user requirements. The operational vision for the StratWar wargame changed little between the first day of the project and the actual ACSC exercise. A major shift in the vision of the wargame would probably have at least delayed the first ACSC exercise by a year.

> Lesson: Reasonably stable requirements can mitigate the risk of a tight software development schedule.

StratWar had an end goal, the ACSC exercise, to guide it. This helped the project team avoid sub-optimization: effort expended to perfect one portion of the project to the detriment of the overall project [14]. McConnell in [13] refers to the problem of sub-optimization as "developer gold-plating." Without this strong goal it might have been easy to start "fixing" the wargaming engine or to better integrate Oracle into the general CRES framework—both of which, while interesting and seemingly beneficial to StratWar, would have delayed overall progress on the project—probably causing the ACSC exercise deadline to be missed. Sub-optimization is an insidious danger because it appears beneficial to the project and provides project members with a feeling of comfortable accomplishment—it is a way for the project team to avoid confronting risk. When, several years after StratWar, I took over the transition of a 4 million SLOC PL/I system to Ada, I was forced to restructure the entire effort and trash an entire year of work by the development team due to a sub-optimization problem. The development

team had spent nearly a year creating several hundred pages of low-level data flow diagrams from the legacy PL/I code using a commercial tool. The creation of data flow diagrams had become a comfortable task that the team members were good at. It had the added benefit that the PL/I maintainers didn't understand nor really care about the low-level diagrams so the development team was rarely forced to interact with them on this task. The idea that the data flow diagrams were simply a tool to understand and document the legacy system had been forgotten. In fact, none of the development team understood much about the overall capabilities of the legacy PL/I system. Worse, instead of using the data flow diagrams to facilitate communication with the maintainers they had used it to shut them out of the work. Sadly, management had not noticed the problem as a steady stream of impressive technical documentation was being produced by the team. StratWar avoided this type of problem—perhaps in part because it was developed with frequent interaction with its users (i.e., the operators and ACSC representatives), but also, I believe, because the visible goal of the ACSC exercise focused the team.

> Lesson: Visible goals can help focus the team on the whole rather than the parts and avoid the serious problem of sub-optimization.

StratWar had an effective development team. The formation of an effective matrix development team was a key reason the project succeeded. DeMarco and Lister in [7] present ideas on how to "grow effective teams." The StratWar team exhibited several of the signs they present as evidence of an effective or "jelled" team: a clear goal, low turnover, a strong sense of identity, and a sense of eliteness. Several environmental and sociological factors allowed the StratWar team to "jell." The climate at the start of the project helped. Everyone involved with CRES was tired of waiting for the contractor to deliver a working wargame and wanted to take action toward that goal. In addition, the team was composed of (mostly) Air Force officers—giving an additional shared culture that helped team formation. As Air Force officers each member of the team was invested in a successful outcome. If StratWar emerged as a success it would further the career of each team member. However, it is critical to note that an effective team didn't appear the day the tasking letter from Col Ware appeared on each of team member's desk. As the narrative of the "boost phase" describes, the initial technical work was often frustrating and chaotic. In my experience, this early chaos ends up benefiting the project—as long as it subsides over time as it did in StratWar. It is a serious warning sign, in my mind, if the first several weeks of a non-trivial software development effort are cordial and ordered. On the other hand, it is also a serious warning sign if chaos continues unabated and progress and technical solutions don't emerge. The development of the StratWar development team as a group is consistent with several models of group development such as Cog's ladder model [16], i.e., polite stage, why

we're here, bid for power, constructive, and esprit, and the Tuckman model [17], i.e., forming, storming, norming, and performing.

The team was talented and was willing to work hard to learn in areas where their knowledge was weak. Without talented people any project, software or otherwise, is doomed to failure no matter how high the motivation and moral of the people. The problem of social loafing [12], the tendency of individual group members to reduce their work effort as the group they belong to increases in size, was avoided by keeping the team size small and by an early assignment of clear roles and responsibilities. As the narrative notes, at many times various team members contributed creative solutions to hard problems that threatened the project—most dramatically illustrated by Capt Atwell and the idea for the color X terminal. The team developed a solid identity around the StratWar project and, to a degree, a sense of eliteness. They had been hand selected by senior management to create a first of its kind wargame. Finally, the StratWar team experienced little personnel turn-over during the development (Lt Col Burns did retire before the first exercise occurred). In subsequent projects, I have found personnel turn-over, especially at key points in the project lifecycle, to be a serious problem. If you start to attend lots of going away lunches—be worried.

> Lesson: Effective teams take time to coalesce. Conflict is a normal part of the teambuilding process. Software practitioners must be open and motivated to learn new knowledge (e.g., technical or domain) all the time.

The StratWar development team confronted project risks. Consider the early identification of the tremendous amount of technical knowledge required to be learned on-the-job by the programmer team. Constraining the design of the software to "limit learning" was an unorthodox approach that successfully mitigated this risk. In several subsequent efforts I have found introspection on the real capabilities of a team to benefit technical design—the most elegant technical design is of no value if your team is not capable of understanding and building it. A better option, when feasible, is to allocate time and funds for training the technical team.

Risk management as a core part of the software development process was proposed by Boehm in [3] as part of his spiral model of software development. However, risk management is often presented in terms of estimating probabilities of loss for identified risks and their potential impact in time or cost [12, 13]. I have noted that "comfortable" work is often low-risk work and driving a team to deal with "uncomfortable" work early in a project acts as an effective risk mitigation approach.

> Lesson: Mitigate risks early in the project—avoid the natural tendency to focus on "comfortable" (i.e., low-risk) work early in the project.

StratWar made effective use of off-the-shelf software. The use of X and Oracle were key to the successful development of the software on schedule. Use of Interleaf increased the quality of the project documentation. Every software development effort I have been involved with since StratWar has made increasing use of off-the-shelf software—it is often the only way a project can be made feasible (i.e., without the engineering savings gained by use of the off-the-shelf product the project would be too costly to develop at all). The StratWar team did a good job of examining the industry situation in its selection of X and Oracle—to their benefit. The lesson here is to ensure that off-the-shelf software choices are deliberate decisions by the team—not technical accidents. A subsequent project I was part of illustrates what can go wrong. This project involved several distinct software development teams in distributed locations. When I joined the project, management had no idea what off-the-shelf products were in use—"whatever the programmers had on their machines" was the management attitude. After discussion with all the teams, I discovered that two C++ compilers, two Ada95 compilers, two design tools, and three scripting languages were being used at the various locations. The cost for this lack of off-the-shelf software management to the project was hundreds of thousands of dollars in training costs and in off-the-shelf software licenses to develop and maintain the software that could have been avoided.

> Lesson: Make sure off-the-shelf software choices are deliberate decisions—not technical accidents.

StratWar had no political "enemies." Unlike many development projects I have subsequently been involved with, the StratWar project had no political "enemies"—groups within the organization that would benefit from, and actively lobby for, the failure of the project. In this category I include only groups within the same organization or company as the development team—not competing companies. Corporate politics, either intentionally (perhaps to foster a sense of internal competition) or unintentionally can create "enemies" of a software development. This can become a serious issue when the software is replacing a current operational system and parts of an organization are worried about the impact of the change on manpower and funding. When political enemies exist, care must be taken to continuously understand and, to a degree, manage senior management perceptions of the project. Honest, frequent, and if possible face-to-face updates to senior management about any potential problems are an effective mitigation—avoid surprises. The StratWar development did not face this problem. Even during the color X terminal crisis no one was really pleased about the crisis and everyone was relieved when an acceptable technical solution was found.

> Lesson: Realize political "enemies" to your project may exist—keep senior management informed about problems and avoid surprises.

What went wrong

The StratWar development team didn't understand senior management's needs. Even in 1990 it was well understood that a difference exists between the user of a software system and its customer [10]. Our users were the AFWC operators and the ACSC students and faculty. Our customers were the AFWC leadership. The overwhelming success of the wargame provides evidence that we listened to our users. However, the sagas of the color X terminal and the "Saturday Test" also provide evidence we didn't listen to our customer—the project survived but the team paid a high price in time and frustration.

At the time of the StratWar development the discipline of software project management did not exist in the Air Force as it does for modern software development projects. As we have seen, 2176A-style document driven development was used—tempered somewhat by judgment. The modern Air Force approach to Project Management Reviews would have helped inform and involve senior management in the StratWar development. As it was, the project manager, as an operator, viewed the software as a black box. Hence, working visible capability was his sole measure of software progress. This is good because working software capability is a real metric and bad because with only this metric cost and schedule are often poorly managed. For StratWar, the functional manager, Lt Col Ellertson, was expected to manage within the black box—however the 2167A approach drove him to document counting augmented by informal discussions of technical work and progress. This approach proved effective for StratWar, but was very ad hoc. A better thought out software development process (as opposed to document driven development) was an emerging idea in practice at this time—the initial textbook on the CMM [11] was published during the StratWar development (as was [3]). However, in its interaction with the users to guide the software development and inclusion of users as part of the development team, i.e., the operators, the StratWar development embraced, to a degree, some of the modern ideas espoused by extreme programming [4].

> Lesson: Work to elicit critical senior management requirements—realize what is important to you may be very different than what is important to them. Use Project Management Reviews, or a similar technique, to inform and stay in contact with senior management.

The contractor architecture of CRES was trusted for StratWar. The architecture and high-level design of StratWar was subject to little analysis because, at the start of the project, it mirrored very closely the contractor approach to CRES—the same basic configuration just with better graphics than the PC could provide. This was a mistake: trusting an untried architecture directly led to the problems of duplicate data during testing of the automatic targeter and, more seriously, the long rocky road to achieving a solid technical solution to orchestrating the wargaming engine during the execution phase. The idea

that a system's software architecture [15] should be designed, documented, and analyzed is better understood today than it was in 1989. Today, standard modeling languages such as UML [9] as well as focused textbooks on software architecture, such as [2] and [6], provide concrete techniques practitioners can use to develop and document a system's software architecture. However, even today, these techniques are not ubiquitous. StratWar serves as a concrete example of the serious technical problems that can emerge when the software architecture of a non-trivial software system is poorly understood and receives little attention.

> Lesson: Understand, document, and analyze your software's architecture.

All the Sun 386i workstations were purchased before the project began. Because the only apparent impact to the project was that the automatic targeter ran somewhat slowly during the ACSC exercise, this could be perceived as a minor problem. However, this is a very serious mistake for any project to make. Hardware purchases should be phased with the software development effort to lower risk—in terms of both invested capital and hardware capability—and to gain greater benefits from Moore's law and competition within the off-the-shelf computer hardware market. StratWar should have purchased only enough Sun 386i workstations for development and initial testing—then later in the project purchased the machines for production use. For StratWar, this approach would have allowed the cheaper purchase of more capable workstations around 14 months into the project. In reality, the 386i workstations were first upgraded to 8MB of memory for the 2nd ACSC exercise (to speed up the automatic targeter) and replaced two years later with SPARCstation 2 workstations.

> Lesson: Purchase off-the-shelf computer hardware as late as possible in a project to lower risk and exploit Moore's law—both in terms of invested capital and hardware capability.

Conclusion

The narrative and analysis of the StratWar wargame development provide a case study of the issues faced by software practitioners during a real-world software development project—even a successful one like StratWar. Expect and anticipate problems, use judgment and teamwork to solve them and learn from past projects, like StratWar, whenever possible.

Acknowledgements

Thanks to Mark Kross and Bart Atwell who, as original members of the StratWar development team, helped out my memory and provided insightful critiques of my analysis. Aaron Greenhouse and Bill Scherlis helped shape the writing into something understandable to an audience who didn't participate in the project.

References

[1] Air Force Wargaming Institute web site at http://www.cadre.maxwell.af.mil/wg

[2] Bass, Clements, and Kazman, *Software Architecture in Practice*. 2nd edition, Addison-Wesley, 2003.

[3] Boehm, *A Spiral Model of Software Development and Enhancement*. IEEE Computer, May 1988, pp 61–72.

[4] Beck, *Extreme Programming Explained*. Addison-Wesley, 2000.

[5] Cooper, *About Face the Essentials of User Interface Design*. IDG, 1995.

[6] Clements, et al. *Documenting Software Architecture: Views and Beyond*. Addison-Wesley, 2003.

[7] DeMarco and Lister, *Peopleware: Productive Projects and Teams*. Dorset House, 1987.

[8] DOD-STD-2167A, *Defense System Software Development*. on-line at http://www2.umassd.edu/SWPI/DOD/MIL-STD-2167A/DOD2167A.html

[9] Fowler, *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. 3rd edition, Addison-Wesley, 2004.

[10] Gause and Weinberg, *Exploring Requirements: Quality before Design*. Dorset House, 1989.

[11] Humphrey, *Managing the Software Process*. Addison-Wesley, 1989.

[12] Karau and Williams, *Social loafing: A meta-analytic review and theoretical integration*. Journal of Personality and Social Psychology, 65(4), 1993, pp 681–706.

[13] McConnell, *Rapid Development: Taming Wild Software Schedules*. Microsoft Press, 1996.

[14] Meredith and Mantel, *Project Management a Managerial Approach*. 4th edition, John Wiley & Sons, 2000.

[15] Shaw and Garlan, *Software Architecture: Perspectives on an Emerging Discipline*, Pearson Education, 1996.

[16] *Squadron Officer School Curriculum* (Maxwell AFB, Ala.: Air University, 1987), 3130 R-1 through R-2 and 3160 R-1 through R-2.

[17] Tuckman, *Developmental sequence in small groups*. Psychological Bulletin, 63, 1965, pp 384–399.

[18] Winograd, *Bringing Design to Software*. Addison-Wesley, 1996.